Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1-20. Canceled

21. (Currently Amended) A method for removing dead code in code fragments of a program, comprising:

identifying each instruction assigning a register that is possibly live for ~~a first~~ each exit in a first code fragment;

storing, in an epilog associated with each exit of the first code fragment, information corresponding to each instruction identified for the corresponding exit;

identifying each register that is assigned before being read in a second code fragment having a first entry;

at a time when linking a ~~the~~ first exit from the first code fragment to the first entry in the second code fragment, comparing the registers in the instructions identified as being possibly live in the first code fragment with the identified registers in the second code fragment; and

eliminating an instruction in the first code fragment based on the comparison.

22. (Previously Presented) A method according to claim 21, wherein an instruction identified in the first code fragment is eliminated if the register assigned in the identified instruction matches a register identified in the second code fragment.

23. (Previously Presented) A method according to claim 21, wherein an instruction assigning a register is possibly live if there is an exit before the register is reassigned and the exit is before the register is read.

24. (Previously Presented) A method according to claim 21, wherein an instruction assigning a register is possibly live if the register is not read subsequently in the first code fragment.

25. (Previously Presented) A method according to claim 21, wherein eliminating the instruction comprises overwriting the instruction with a NOP.

26. (Previously Presented) A method according to claim 21, wherein eliminating the instruction comprises compacting the surrounding instructions to delete the eliminated instruction.

27. (Currently Amended) A method for removing dead code in code fragments of a program ~~according to claim 21, further~~ comprising:

identifying each instruction assigning a register that is possibly live for a first exit in a first code fragment;

generating a first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction identified in the first code fragment; ~~and~~

identifying each register that is assigned before being read in a second code fragment having a first entry;

generating a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is one of the identified registers in the second code fragment;

at a time when linking the first exit from the first code fragment to the first entry in the second code fragment, comparing the registers in the instructions identified as being possibly live in the first code fragment with the identified registers in the second code fragment; and

eliminating an instruction in the first code fragment based on the comparison.

28. (Previously Presented) A method according to claim 27, where said eliminating step includes eliminating an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

29. (Canceled)

30. (Currently Amended) A method according to claim 21~~9~~, further comprising:

identifying each register that is assigned before being read after each entry in the second code fragment;

storing, in a prolog associated with each entry of the second code fragment, information corresponding to each register identified for the corresponding entry.

31. (Previously Presented) A method according to claim 30, wherein the information in the corresponding epilogs of the exits in the first code fragment and the information in the corresponding prologs of the entries in the second code fragment are stored prior to the linking of the first exit from the first code fragment to the first entry in the second code fragment.

32. (Currently Amended) A computer readable medium operable on a computer for removing dead code in code fragments of a program, the computer readable medium configured to:

identify each instruction assigning a register that is possibly live for ~~a first~~ each exit in a first code fragment;

store, in an epilog associated with each exit of the first code fragment, information corresponding to each instruction identified for the corresponding exit;

identify each register that is assigned before being read in a second code fragment having a first entry;

at a time when linking ~~the~~ a first exit from the first code fragment to the first entry in the second code fragment, compare the registers in the instructions identified as being

possibly live in the first code fragment with the identified registers in the second code fragment; and

eliminate an instruction in the first code fragment based on the comparison.

33. (Previously Presented) A computer readable medium according to claim 32, wherein an instruction identified in the first code fragment is eliminated if the register assigned in the identified instruction matches a register identified in the second code fragment.

34. (Previously Presented) A computer readable medium according to claim 32, wherein an instruction assigning a register is possibly live if there is an exit before the register is reassigned and the exit is before the register is read.

35. (Previously Presented) A computer readable medium according to claim 32, wherein an instruction assigning a register is possibly live if the register is not read subsequently in the first code fragment.

36. (Previously Presented) A computer readable medium according to claim 32, further configured to overwrite the instruction with a NOP to delete the eliminated instruction.

37. (Previously Presented) A computer readable medium according to claim 32, further configured to compact the surrounding instructions to delete the eliminated instruction.

38. (Currently Amended) A computer readable medium ~~according to claim 32, further configured to:~~ operable on a computer for removing dead code in code fragments of a program, the computer readable medium configured to:

identify each instruction assigning a register that is possibly live for exit in a first code fragment;

generate a first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction identified in the first code fragment; ~~and~~

identify each register that is assigned before being read in a second code fragment having a first entry;

generate a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is one of the identified registers in the second code fragment;

at a time when linking ~~the~~ a first exit from the first code fragment to the first entry in the second code fragment, compare the registers in the instructions identified as being possibly live in the first code fragment with the identified registers in the second code fragment; and

eliminate an instruction in the first code fragment based on the comparison.

39. (Previously Presented) A computer readable medium according to claim 38, further configured to eliminate an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

40. (Canceled)

41. (Currently Amended) A computer readable medium according to claim 32~~40~~, further configured to:

identify each register that is assigned before being read after each entry in the second code fragment;

store, in a prolog associated with each entry of the second code fragment, information corresponding to each register identified for the corresponding entry.

42. (Previously Presented) A computer readable medium according to claim 41, wherein the information in the corresponding epilogs of the exits in the first code

fragment and the information in the corresponding prologs of the entries in the second code fragment are stored prior to the linking of the first exit from the first code fragment to the first entry in the second code fragment.